18. Netwerken

Niet voor de ESP32-C3, wel de andere ESP's

De controllers van Espressif hebben hard- en software aan boord voor de verbinding met een netwerk. De hardware is een WiFi module, de software zit in **RTOS**, het besturingssysteem dat elke ESP-controller heedft. RTOS bestuurt o.a. de netwerktoegang. De netwerkfuncties van MicroPython zijn een de verbinding tussen onze programma's en de netwerkfuncties van RTOS. Een voordeel van de dubbel-core ESP's is dat één processor zich aan het netwerk kan wijden, de andere processor voert onze programma's uit. De Raspberry Pi Pico W heeft een WiFi module aan boord, die doet al het werk voor de netwerktoegang.

In de bijlage is een paragraaf gewijd aan de basisbeginselen van een TCP/IP netwerk.

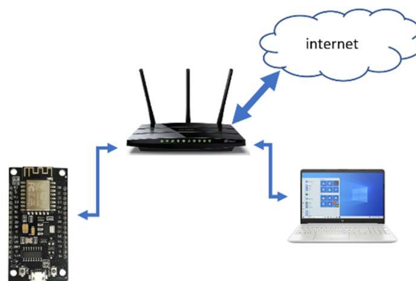
De verbinding met het netwerk

We kunnen een ESP controller op twee manieren met een netwerk verbinden:

- Als station: de controller maakt verbinding met een router of access-point
- Als access-point: andere toestellen maken verbinding met de controller

Verbinding met een router/access-point

De figuur hieronder toont een typisch huisnetwerk: De router vormt de verbinding tussen het lokale huisnetwerk met Internet. De router zorgt ook voor de juiste netwerkinstellingen van de aangesloten toestellen. De toestellen op het netwerk zijn verbonden met de router via een netwerkkabel of draadloos. Een draadloos netwerk heeft een naam, de **ssid**. Meestal is dit netwerk beveiligd met een paswoord.



Figuur 105: een typisch huisnetwerk

Module network heeft de nodige methodes voor de netwerktoegang:

- **network.WLAN(network.STA_IF)** creëert een WLAN object voor verbinding met de router

- **active(True)** en **active(False)** activeren of de-activeren de netwerkinterface
- **active()** geeft de toestand van de interface
- **connect (ssid, password)** maakt verbinding met het draadloos netwerk. De DHCP-serv er van de router levert automatisch de IP-parameters aan de controller.
- **isconnected()** geeft aan of we verbonden zijn
- **ifconfig()** geeft de IP-parameters van de verbinding.

De verbinding met connect lukt niet altijd direct. Meestal moeten we enkele keren proberen. In het voorbeeld maken we verbinding met netwerk “WLAN-dirk” met paswoord “1234”

```
#-----------------------------#
#  wifi-sta.py                #
#                             #
#  16 januari 2022            #
#-----------------------------#
import network, time
essid = "WLAN-dirk"
passwd = "1234"
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
```

Het resultaat is ('192.168.1.47', '255.255.255.0', '192.168.1.1', '192.168.1.1'), het IP-adres van de module, het subnet-mask, het adres van de router en van de DNS-server. We kunnen dit controleren met opdracht **ping** op de PC vanaf de opdracht prompt.

```
Opdrachtprompt

C:\Users\dirkg>ping 192.168.1.47

Pinging 192.168.1.47 with 32 bytes of data:
Reply from 192.168.1.47: bytes=32 time=82ms TTL=255
Reply from 192.168.1.47: bytes=32 time=4ms TTL=255
Reply from 192.168.1.47: bytes=32 time=5ms TTL=255
Reply from 192.168.1.47: bytes=32 time=13ms TTL=255

Ping statistics for 192.168.1.47:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 82ms, Average = 26ms

C:\Users\dirkg>
```

Figuur 106: opdracht *ping*

Zelf IP-parameters indgeven kan ook, we gebruiken methode ifconfig:

netSTA.ifconfig(('192.168.1.57', '255.255.255.0', '192.168.1.1', '192.168.1.1')) let op, we gebruiken dubbele haken.

Opmerking:

Het is geen goed idee om jouw netwerkgegevens zichtbaar in een programma op te nemen. Vroeg of laat ga je jouw programma's doorgeven of publiceren. Niet iedereen moet toegang krijgen tot jouw persoonlijk WiFi netwerk. We maken daarom een bestand **secrets.py** aan met jouw geheime gegevens. Dat bestand kopiëren we naar de controller. Gebruikers van jouw programma gebruiken hun eigen versie van secrets.py. De bestanden worden dan:

```
#------------------------------#
#  secrets.py                  #
#                              #
#  2 augustus 2022             #
#------------------------------#
secrets = {
    'ssid' : 'WLAN-dirk',
    'pw' : '1234'
}
```

```
#------------------------------#
#  wifi-sta.py                 #
#                              #
#  2 augustus 2022             #
#------------------------------#
import network, time
from secrets import secrets
essid = secrets['ssid']
passwd = secrets['pw']
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
```

Netwerktoepassingen

De toepassingen in een netwerk zijn server-client gebaseerd: een client stuurt een vraag naar de server en de server stuurt een antwoord. Server en cliënt gebruiken dezelfde netwerkpoort en hetzelfde protocol (UDP of TCP). Zie ook de bijlage. Voor deze verbinding gebruiken we klasse socket van **module socket**. Enkele methodes hiervan zijn:

- **addr = socket.getaddrinfo(server, poort)** geeft adresinfo van de server. Server kan je opgeven met zijn url-adres ("www.google.be") of zijn IP-nummer.

- **s = socket.socket()** maakt een socket object
- **connect()** maakt een TCP-verbinding met de server
- **send()** stuurt een bericht over het netwerk
- **rcv()** leest een bericht uit het netwerk

Gegevens van een webserver vragen

Een webserver gebruikt poort 80 en een TCP-verbinding. TCP is een betrouwbaar protocol, met foutdetectie en connectie. In het voorbeeld lezen we 1000 bytes uit webserver www.google.be.

```
#-----------------------------#
#  webclient.py               #
#                             #
#  2 augustus 2022            #
#-----------------------------#
import network, time
from secrets import secrets
essid = secrets['ssid']
passwd = secrets['pw']
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
import socket
servernaam = "www.google.be"
addr = socket.getaddrinfo(servernaam, 80)[0][-1]
print (addr)
s = socket.socket()
s.connect(addr)
s.send(b'GET / HTTP/1.1\r\nHost: '+servernaam+'\r\n\r\n')
data = s.recv(1000)
print(data)
s.close()
```

- Het programma gebruikt ook secrets.py, die bevat de ssid en paswoord van jouw WiFi netwerk.
- Het eerste deel van het programma is de configuratie van WiFi op de controller uit vorig voorbeeld.
- Het resultaat van getaddrinfo is een tuple waar we één element van nodig hebben: element [0][-1]
- Send stuurt een GET naar de server. Hiermee wordt informatie gevraagd in HTTP-vorm. Dat is webserver-jargon.
- Recv(1000) leest de eerste 1000 tekens van het antwoord in.

Het resultaat is een stuk webpagina, geschreven in HTML-taal. In de bijlage gaan we daar iets meer van bekijken.

```
>>> %Run -c $EDITOR_CONTENT
 ('192.168.1.57', '255.255.255.0', '192.168.1.1', '192.168.1.1')
 ('142.250.179.163', 80)
 b'HTTP/1.1 200 OK\r\nDate: Mon, 17 Jan 2022 14:07:08 GMT\r\nExpires: -1\r\nCache-Control: private, max-
 age=0\r\nContent-Type: text/html; charset=ISO-8859-1\r\nP3P: CP="This is not a P3P policy! See g.co/p3p
 help for more info."\r\nServer: gws\r\nX-XSS-Protection: 0\r\nX-Frame-Options: SAMEORIGIN\r\nSet-Cookie
 : NID=511=KROq86bOhiN8MMpaVwV1d4sfgnu_8-c8HegrCE0tV7IMlMpzTMCTRzL7sA--BsuXQCsZ3m-VN1MihxkpEybjbvXctUwD8
 UA9aYKhNV66ptuWGxXNNcJktZFj0kQWX8sBhPvn6_Edsq7aIVLspCV-etpg6Kl8Uq8m2DNQxFqk3ro; expires=Tue, 19-Jul-202
 2 14:07:08 GMT; path=/; domain=.google.be'
>>>
```

Figuur 107: webclient

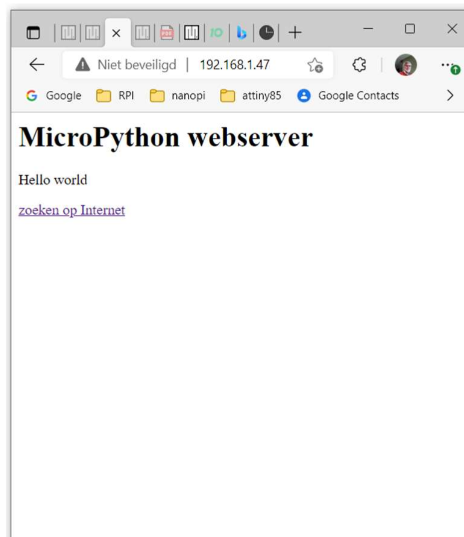
Een webserver

Het programma hieronder is een heel eenvoudige webserver.

```
#-----------------------------#
#  webserver.py               #
#                             #
#  2 augustus 2022            #
#-----------------------------#
import network, time, usys
import socket
from secrets import secrets
essid = secrets['ssid']
passwd = secrets['pw']
pagina = """<html><head><title>voorbeeldpagina</title></head>
<body><h1>MicroPython webserver</h1>
<p>Hello world</p>
<a href="http://www.google.be">zoeken op Internet</a>
</body></html>"""
print (pagina)
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
netSTA.ifconfig(('192.168.1.59', '255.255.255.0', '192.168.1.1', '192.168.1.1'))
print(netSTA.ifconfig())
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
print('listening on', addr)

while True:
    conn, addr = s.accept()
    print('client connected from', addr)
    response = pagina
    conn.send(response)
    conn.close()
```

- Het eerste deel maakt de verbinding met het WiFi netwerk, analoog aan vorig voorbeeld.
- Het tweede deel definieert een socket met poort 80, de webserver poort. Het stuk in de while True lus gaat na of er een aanvraag is van een webclient en stuurt de webpagina terug.
- De opdracht print(netSTA.ifconfig()) toont het IP-adres van de server. Dat geven we in de browser van de PC in. Handiger is een vast IP-adres voor de server, gebruik daarvoor instructie netSTA.ifconfig(), zie hierboven.
- String pagina bevat een stuk HTML code, dat wordt de webpagina. Hierin staat een titel, boodschap 'Hello" en een link naar Google.be.



Figuur 108: eenvoudige webserver

Toepassing: GPIO's uitlezen met de Webserver

De webserver stuurt een pagina door met HTML-code. Daar kan je alle soorten gegevens inzetten zoals de toestand van een poort, de waarde van een sensor enz. Het voorbeeld hieronder leest een temperatuursensor uit en toont het resultaat op de webpagina.

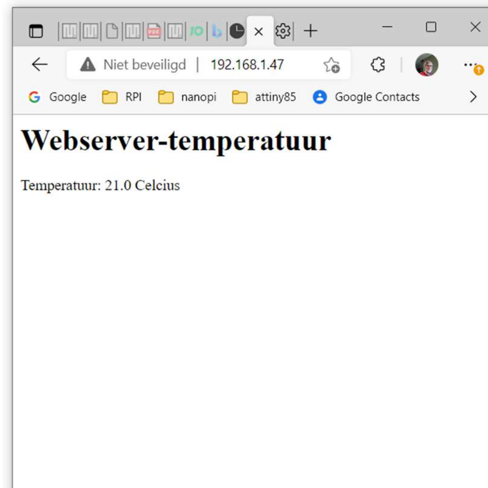
We meten de temperatuur een DS18B20-sensor aan GPIO4. Het eerste deel van het programma is afgeleid uit het programma voor deze sensor uit hoofdstuk *Sensoren*. De webpagina bestaat uit een stuk HTML-code. Daar is variabele tempe in opgenomen, de uitlezing van de sensor. De rest van het programma is identiek aan de webserver uit vorig voorbeeld. Helaas, deze configuratie drukt symbool ° voor voor de weergave van de temperatuur niet goed af, vandaar de ongewone notatie 21.0 Celsius.

```
#-----------------------------#
#  webserver-temp.py          #
#                             #
#  2 augustus 2022            #
#  Dirk Ghysels               #
#-----------------------------#
import network, time, usys, onewire, ds18x20
from secrets import secrets
essid = secrets['ssid']
passwd = secrets['pw']
dsPin = machine.Pin(4)
ow = onewire.OneWire(dsPin)
sensor = ds18x20.DS18X20(ow)
roms = sensor.scan()
print(roms)
sensor.convert_temp()
for rom in roms:
        tempe = str(sensor.read_temp(rom))
print ("T = ", tempe)
pagina = """<html><head><title>temperatuur
server</title></head>
<body><h1>Webserver-temperatuur</h1>
<p>Temperatuur: """
pagina += tempe
pagina += " Celcius </p></body></html>"
print (pagina)
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())

import socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)

print('listening on', addr)

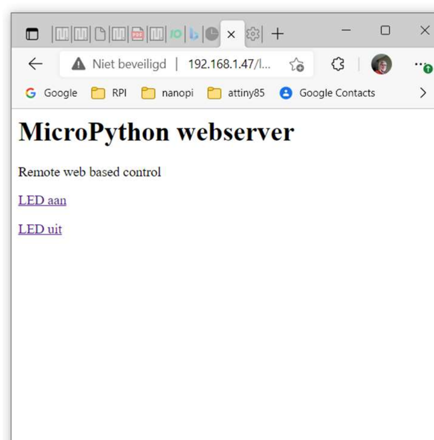
while True:
    conn, addr = s.accept()
    print('client connected from', addr)
    response = pagina
    conn.send(response)
    conn.close()
```



Figuur 109: uitlezen temperatuur met webserver

Toepassing: schakelen met een webserver

In het voorbeeld gaan we met de webserver een LED aan of uitschakelen.



Figuur 110: schakelen met een webserver

String pagina, een string van 7 regels, is de webpagina. Hierin zie je 2 hyperlink. Led aan verwijst naar `http://192.168.1.47/led=aan`, `dat is hetzelfde adres met een stukje "/led=aan"  De server geeft dezelfde pagina terug. We gaan na of dat stukje voorkomt in request, de aanvraagstring van de client. Indien ja, laten we de LED branden.`

```
#-----------------------------#
#  webserver-LED.py           #
#                             #
#  2 augustus 2022            #
#  Dirk Ghysels               #
#-----------------------------#
import network, time, usys
from machine import Pin
led = Pin(2, Pin.OUT)
led.value(1)
from secrets import secrets
essid = secrets['ssid']
passwd = secrets['pw']
pagina = """<html><head><title>voorbeeldpagina</title></head>
<body><h1>MicroPython webserver</h1>
<p>Remote web based control</p>
<p><a href="http://192.168.1.47/led=aan">LED aan</a></p>
<p><a href="http://192.168.1.47/led=uit">LED uit</a></p>
<p></p>
</body></html>"""
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
import socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
while True:
    conn, addr = s.accept()
    print('client connected from', addr)
    request = conn.recv(1024)
    if ("/led=aan" in request):
        led.value(0)
    else:
        led.value(1)
    print("led : ", led)
    response = pagina
    conn.send(response)
    conn.close()
```

opmerking:

In het voorbeeld is de LED aangesloten tussen GND en GPIO2, hij brandt als die waarde 0 heeft.

Het adres van de server staat 2 maal in string pagina. Pas dat aan, jouw server heeft waarschijnlijk een ander adres.

Toepassing: een internetklok

In hoofdstuk Timers en klokken hebben we gezien dat de tijd wordt bijgehouden als de **epoch**, het aantal seconden sinds 1 januari 1970, 0:00:00, UTC. Deze tijd is de tijd in Greenwich. Embedded systemen, zoals de controllers voor MicroPython gebruiken een epoch vanaf 1 januari 2000. Hieruit berekenen we de tijd in uren, minuten, …

Op Internet zijn er NTP-servers (Network Time Protocol) die de de tijd uitzenden. We kunnen die gebruiken om de RTC van ons systeem te synchroniseren. De NTP-organisatrie raadt aan om hiervoor server www.ntppool.org te gebruiken. Als we die uitlezen met programma webclient van hierboven krijgen we volgend resultaat:

```
Shell
>>>
>>> %Run -c $EDITOR_CONTENT
 ('192.168.1.47', '255.255.255.0', '192.168.1.1', '192.168.1.1')
 ('151.101.38.137', 80)
 b'HTTP/1.1 301 Moved Permanently\r\nServer: Varnish\r\nRetry-After: 0\r\nLocation: https://www.ntppool.
 org/\r\nContent-Length: 0\r\nAccept-Ranges: bytes\r\nDate: Wed, 19 Jan 2022 14:42:06 GMT\r\nVia: 1.1 va
 rnish\r\nConnection: close\r\nX-Served-By: cache-ams21034-AMS\r\nX-Cache: HIT\r\nX-Cache-Hits: 0\r\nX-T
 imer: S1642603326.210979,VS0,VE0\r\nStrict-Transport-Security: max-age=12096000\r\n\r\n'
>>>
```

Figuur 111: uitlezing van een ntp-server

Datum en tijd kan je direct aflezen: woensdag 19 januari 2022, 14:42:06. Deze tijd is Greenwich tijd, in West-Europa is het één uur later. De epoch, het aantal seconden, staat er ook in, dat vind je in het stuk *Timer: S1642603326.210979*,. De cijfers tussen S en punt hebben we nodig. We kunnen ze omzetten naar een normale tijd met functie

time.localtime(1642603326)

Het resultaat is

(2052, 1, 19, 14, 42, 6, 4, 19),

Dat is 19 januari 2052, 14:42:06. Let op, NTP telt vanaf 1970, de controller vanaf 2000, we moeten er 30 jaar van aftrekken en één uur bijtellen om datum en tijd in België of Nederland terug te vinden. Tijdens de zomertijd tel je 2 uur bij.

In het programma moeten we de epoch uit het antwoord van de ntp-server peuteren. Daarvoor gebruiken we functie index():

String.index("woord")

geeft de plaats waar voor het eerst woord voorkomt in de string.

Daarna zoeken we vanaf die plaats naar een opeenvolging van cijfers, dat is de epoch. Uit de epoch halen we uur, minuut, seconde, dag, maand en jaa. We zetten alles in de RTC, rekening houdend met de nodige correcties. De rest van het programma leest de RTC en zet alles op het scherm. Het resultaat is een klok op het scherm die volledig gelijk loopt met een een DCF77, een radiogestuurde klok.

```
#------------------------------#
#  ntp-client.py               #
#                              #
#  3 augustus 2022             #
#------------------------------#
import network, time
from machine import RTC
from secrets import secrets

#   tz = 1  # tijdzonecorrectie West Europa, wintertijd
tz = 2  # tijdzonecorrectie West Europa, zomertijd
rtc = machine.RTC()
essid = secrets['ssid']
passwd = secrets['pw']
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
import socket
servernaam = "www.ntppool.org"
addr = socket.getaddrinfo(servernaam, 80)[0][-1]
print (addr)
s = socket.socket()
s.connect(addr)
s.send(b'GET / HTTP/1.1\r\nHost: '+servernaam+'\r\n\r\n')
data = str(s.recv(10000))
print(data)
s.close()
cijfers = ('0','1','2','3','4','5','6',’7’,'8','9')
epochstr = ""
plTimer = data.index("nX-Timer")
print (plTimer)
k = plTimer + 7

while (data[k] not in cijfers):
    k = k+1
while (data[k] in cijfers):
    epochstr += data[k]
    k = k+1
print(epochstr)
tijd = time.localtime(int(epochstr))
print(tijd)
rtc.datetime(((tijd[0]-
30),tijd[1],tijd[2],1,(tijd[3]+tz),tijd[4],tijd[5],0))

sec0 = 0
while True:
    nu = rtc.datetime()
    jaar = nu[0]
    maand = nu[1]
```

```
    dag = nu[2]
    uur = nu[4]
    minuut = nu[5]
    seconde = nu[6]
    if (seconde != sec0):
        print("%02u:%02u:%02u %02u/%02u/%u" %(uur, minuut, seconde,
dag, maand, jaar))
        sec0 = seconde
```

Uitbreidingen:

- Probeer zelf een mooi display aan te sluiten
- Variabele **tz** (bovenaan in het programma) is de correctie voor de tijdzone en de zomertijd. Probeer die laatste te automatiseren: zomertijd gaat van de derde zondag van maart tot de derde zondag van oktober.

Module ntptime

Het programma hierboven is een illustratie van netwerktoepassingen in MicroPython en toont hoe we ntp-kunnen gebruiken voor de constructie van een klok. Voor praktische toepassingen gebruiken we klasse ntptime. Die stelt de RTC in en regelt zelf de overgangen van zomer- naar wintertijd. We gebruiken die met

```
import ntptime
ntptime.datetime()
```

Module ntptime werkt niet op een RP Pi Pico W.

Volgend programma gebruikt de klasse voor een klok met een OLED-scherm.

```
#------------------------------#
#  ntp-client-3.py             #
#  esp8285                     #
#                              #
#  3 februari 2022             #
#  Dirk Ghysels                #
#------------------------------#

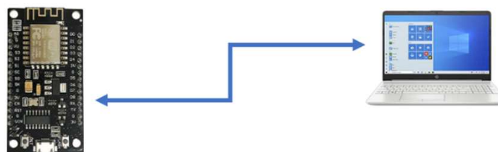
import network, time, ntptime
import machine
from secrets import secrets
from machine import Pin, I2C
import ssd1306
tz = 1  # tijdzonecorrectie West Europa, wintertijd
#  tz = 2  # tijdzonecorrectie West Europa, zomertijd
#esp8285
i2c = I2C(scl=Pin(5), sda=Pin(4), freq = 100000)
print(i2c.scan())
display = ssd1306.SSD1306_I2C(128, 64, i2c)
display.text('ntp-klok', 0, 0, 1)
display.text('(c) Dirk', 0, 20, 1)
display.show()
time.sleep(1)
rtc = machine.RTC()
```

```
essid = secrets['ssid']
passwd = secrets['pw']
netSTA = network.WLAN(network.STA_IF)
netSTA.active(True)
while not netSTA.isconnected():
    netSTA.active(True)
    netSTA.connect(essid, passwd)
    print("connecting to ", essid)
    time.sleep(0.2)
print(netSTA.ifconfig())
ntptime.settime()
nu = rtc.datetime()
print(nu)
sec0 = 0
while True:
    #ntptime.settime()
    nu = rtc.datetime()
    jaar = nu[0]
    maand = nu[1]
    dag = nu[2]
    uur = nu[4] + tz
    if (uur>23):
    uur -= 24
    minuut = nu[5]
    seconde = nu[6]
    if (seconde != sec0):
        print("%02u:%02u:%02u %02u/%02u/%u" %(uur, minuut, seconde,
dag, maand, jaar))
        lijn1 = "%02u:%02u:%02u " %(uur, minuut, seconde)
        lijn2 = "%02u/%02u/%u" %(dag, maand, jaar)
        display.fill(0)
        display.text(lijn1, 0, 0, 1)
        display.text(lijn2, 0, 20, 1)
        display.show()
        sec0 = seconde
```

Opmerking: de library is gebaseerd op UTC-tijd en houdt geen rekening met zomer/wintertijd. Variabele tz maakt de correctie. Tweemaal per jaar moeten we het programma aanpassen.

De controller als access-point

In deze paragraaf werkt de controller als access-point. Die wordt het centrum van een lokaal WiFi-netwerk. PC's en smartphones kunnen hierop inloggen, ze krijgen hun IP-adres en kunnen een website op de controller raadplegen. Toepassingen van de voorbeelden hierboven kunnen we hier ook implementeren.



Figuur 112: de controller als access-point

We gebruiken volgende methodes:

- **network.WLAN(network. PA_IF)** creëert een WLAN object voor de controller als access-point
- **active(True)** en **active(False)** activeren of de-activeren de netwerkinterface
- **active(True)** maakt de interface actief
- **connect (ssid, password)** geeft het draadloos netwerk een ssid en een paswoord. De DHCP-server van de controller levert IP-parameters aan toestellen die zich aansluiten.
- **ifconfig()** geeft de IP-parameters van de controller. Het IP adres van de controller gebruik je om de webpagina op te roepen.

De rest van het programma is analoog aan de voorbeelden van webservers hierboven.

```
#------------------------------#
#  WiFi-AP.py                  #
#                              #
#  23 januari 2022             #
#------------------------------#
import network, socket

ssid='esp8266-AP'
password='123456789'
netAP = network.WLAN(network.AP_IF)
netAP.active(True)
netAP.config(essid=ssid, password=password)
#netAP.connect()
while netAP.active() == False:
  pass
print(netAP.ifconfig())
print(netAP.active())


html = """<html><head><meta name="viewport" content="width=device-
width, initial-scale=1"></head>
<body><h1>Hello World!</h1></body></html>"""

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
  conn, addr = s.accept()
  print('Got a connection from %s' % str(addr))
  request = conn.recv(1024)
  print('Content = %s' % str(request))
  response = html
  conn.send(response)
  conn.close()
```



Figuur 113: het access-point is beschikbaar

Het volgende voorbeeld is de combinatie van een access-point en een webserver. Het is gemaakt voor de RP PI Pico W. In de declaratie van led wordt “LED” gebruikt, dat verwijst naar de ingebouwde led van de Pico W. Verander deze opdracht als je een ander bordje gebruikt.

```
#------------------------------#
#  AP-server.py                #
#                              #
#  8 augustus 2022             #
#------------------------------#
import socket
import network
import machine

ssid = 'AP-Dirk'
password = '123456789'

led = machine.Pin("LED",machine.Pin.OUT)

ap = network.WLAN(network.AP_IF)
ap.config(essid=ssid, password=password)
ap.active(True)

while ap.active() == False:
  pass

print('Connection successful')
print(ap.ifconfig())

html = """<!DOCTYPE html>
<html>
    <head> <title>MicroPython access-point</title> </head>
    <body> <h1>MicroPython access-point</h1>
```

```
        <p>Groetjes vanuit ons nieuw access-point</p>
    </body>
</html>
"""

addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)

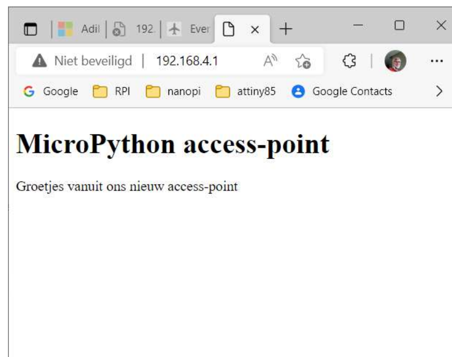
print('listening on', addr)
led.off()

# Listen for connections
while True:
    try:
        cl, addr = s.accept()
        print('client connected from', addr)
        request = cl.recv(1024)
        led.on()
        print(request)

        cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        cl.send(html)
        cl.close()
        led.off()

    except OSError as e:
        cl.close()
        print('connection closed')
```

In regel ***print(ap.ifconfig())*** toont het programma zijn IP-gegevens, die kan de client gebruiken om de webpagina op te roepen.



Figuur 114: MicroPython access-point